A survey of codes and algorithms used in NERSC chemical science allocations

Lin-Wang Wang
NERSC System Architecture Team
Lawrence Berkeley National Laboratory

We have analyzed the codes and their usages in the NERSC allocations in chemical science category. This is done mainly based on the ERCAP NERSC allocation data. While the MPP hours are based on actually ERCAP award for each account, the time spent on each code within an account is estimated based on the user's estimated partition (if the user provided such estimation), or based on an equal partition among the codes within an account (if the user did not provide the partition estimation). Everything is based on 2007 allocation, before the computer time of Franklin machine is allocated. Besides the ERCAP data analysis, we have also conducted a direct user survey via email for a few most heavily used codes. We have received responses from 10 users. The user survey not only provide us with the code usage for MPP hours, more importantly, it provides us with information on how the users use their codes, e.g., on which machine, on how many processors, and how long are their simulations? We have the following observations based on our analysis.

(1) There are 48 accounts under chemistry category. This is only second to the material science category. The total MPP allocation for these 48 accounts is 7.7 million hours. This is about 12% of the 66.7 MPP hours annually available for the whole NERSC facility (not accounting Franklin). The allocation is very tight. The majority of the accounts are only awarded less than half of what they requested for. There is a tremendous demand for capacity computation at NERSC.

(2) There are 56 different codes mentioned in the ERCAP request. The total number of codes is slightly larger than the total number of (groups) accounts. However, since different group might use the same code, thus in average, each group uses 2.2 codes, and each code is used by 1.8 groups. But two third of the codes are used only by one group. The mostly widely used code is Gaussian, used by 11 groups. This is a commercial quantum chemistry code, with a broad collection of functionalities. There is a big overlap of the codes used in chemical science category and the material science category. For example, the following popular codes are used by both categories: VASP, DL_POLY, PWSCF, NWchem, NAMD. There are also common methods used by both categories, most noticeable: density functional theory and classical dynamics.

(3) Surprisingly, the code which uses the most time is S3D, a real space regular grid finite difference code for fluid dynamics to simulate combustion. It uses 2.5 million hours. Partly this is because the INCITE program. S3D does not look like traditional chemistry methods. The next mostly used method in terms of computer time is the conventional quantum chemical codes using Gaussian basis sets to solve Hartree-Fock, MP2, coupled cluster equations. This includes codes like:

Gaussian, Molpro, GAMESS, NWchem, Q-Chem, etc. This traditional quantum chemistry method is closely followed (sometime might be surpassed by) the density functional theory (DFT) method. Nowadays, in all the above traditional quantum chemistry codes, DFT method is also implemented. DFT method is also carried out using the planewave basis (like in the majority of the material science codes). This includes: VASP, PWSCF, DACAPO, CPMD, CP2K etc. The next method in terms of computer time usage is the quantum Monte Carlo method. Unlike the major quantum Monte Carlo codes in material science category, which is based on planewave basis, in quantum chemistry, the major codes are based on Gaussian or other atomic basis set, like in the code Zori. Another category of method which uses a lot of computer time is classical molecular dynamics. This is represented by codes like: DL_POLY, WaterMD, LAMMPS and NAMD. Finally there are many other codes which are focused on one particle problem or using special techniques, for the example: Fouratoms, Fourbody, QDSO, Matrix, etc.

(4) From the direct user survey, we found that the calculations based the traditional quantum mechanical methods (e.g, Gaussian, Q-Chem, Molpro) use small number of processors (from serial to 16 processors). One reason is that they want to calculate many atomic configurations (e.g. to map out the potential manifold in a chemical reaction) for small molecules. Another reason is that those codes do not scale well beyond 16 or 32 processors. The quantum Monte Carlo simulations can use very large number of processors because they are embarrassingly parallel, essentially without communications between different processors. The classical molecular dynamics simulation can also employ large number of processors. Overall, we see a stronger demand for capacity calculation (e.g., turn around time) to capability calculation in the chemical science category.

Table.I: This table lists all the codes and their accumulated (from different groups) MPP hours and number of groups (accounts) using the code. It also give a brief narrative for what does a code do.

Index	N-group	MPP(KH)	Code Name	Narrative
1	3	2,500	S3D	Regular grid fluid dynamics for combustion
2	3	695	Zori	Quantum Monte Carlo, Guassian basis
3	5	519	MOLPRO	High level Q chem. For small system, Gauss.
4	1	500	DACAPO	Planewave DFT, CP, period, ultra-soft,FFT
5	11	409	GAUSSIAN	Gaussian basis Q chem., commercial
6	6	397	CPMD	Planewave DFT
7	5	372	VASP	Planewave DFT, ultra-soft
8	7	364	GAMESS	Gaussian contracted basis, Q-Chem.
9	1	275	WATERMD	Class. MD. For water.
10	6	260	NWCHEM	Q. Chem., many methods, Gaussian, PW
11	4	250	DL_POLY	General Classical MD
12	1	200	PWSCF	Planewave DFT
13	3	194	Q-CHEM	Q Chem. Code with Gaussian contracted basis
14	2	185	LAMMPS	Classical MD

15	1	150	QD_YAEHMPO	DFT QM/MM code
16	1	150	DTMS	Eddy simulation for turbulence.
17	2	125	CP2K	MD and Monte Carlo atom, DFT+class. force
18	1	100	SCHPACK	Time dependent Q dynamics, charge transfer
19	1	81	FLAPW	Augmented PlaneWave DFT
20	1	72	FOURATOM	QM four atom problem with 6-D freedom
21	1	63	GRMD	MD and Monte Carlo atom, DFT+class. Force
22	1	55	FDTD	Maxwell's equation, real space grid, and time
23	1	52	CFRFS	Fluid dynamics, Navier-Stokes equation
24	1	40	MPQC	Massively parallel quantum chemistry
25	1	40	MATRIX	Multi-channel QM, integrel-diff. equation
26	1	27	NUMEROV	Numerov propagation of multi-chann. Wave.
27	1	27	SEMICLASS	Classical wavefunction propagation
28	1	27	SURFACE	6D Schrodinger equation
29	1	25	CEO	Convert trans. Density matrix to polar. tensor
30	1	25	MOLDEN	Gaussian output visualization
31	1	23	PES_FIT	Large list square fitting program
32	1	22	H3	Large scale eigen value problem
33	1	20	RELLIP	Laser photon field propagation
34	1	19	MD	In house MD program for oxide/electrolyte
35	2	10	NAMD	Classical MD program
36	1	10	DALTON	Quantum Chemistry code for molecules
37	1	10	MOLCAS	Quantum Chemistry code with geom. Relax.
38	1	8	UK_R_MATRIX	R-matrix Q chem. multichannel, molecules
39	1	5	FEM_R_MATRI	Electron-molecule scattering problem
40	1	5	QDSO	Exciton, Q dynamics, density matr. Propagat.
41	1	4	DVR_PI	Double photoionization of molecule
42	1	4	FDIFF_ECS	2D Schrodinger's eq. Sparse linear problem
43	1	4	HYBRID_BOUD	Hybrid basis: Gaussian+spherical, Q-Chem.
44	1	4	WAVEPROP	Atom ionization wavefunc. Time propagat.
45	1	4	XKIEN	CI for atomic double photoionization
46	1	4	XMOLTWO_D	Photoionization of hydrogen molecule
47	1	3	ALPS	Code framework for strongly correlated sysm.
48	1	3	AMBER	Classical force field MD
49	1	3	CHARMM	Classical force field MD
50	1	3	XRAYSOS	Third order response function
51	1	3	SDP	Schrodinger's Eq. with on semidefinite prog.
52	1	2	FOURBODY	Four body Fermion on harmonic trapping
53	1	2	MCWF	Time propagation of QM wavefunction
54	1	0	CHEBYSHEV	S-matrix calc. Using chebyshev propagator
55	1	0	ATSP2K	LSJ approx. on QM wave. Atomic property
56	1	0	GRASPVU	Atomic structure with Dirac Eq.

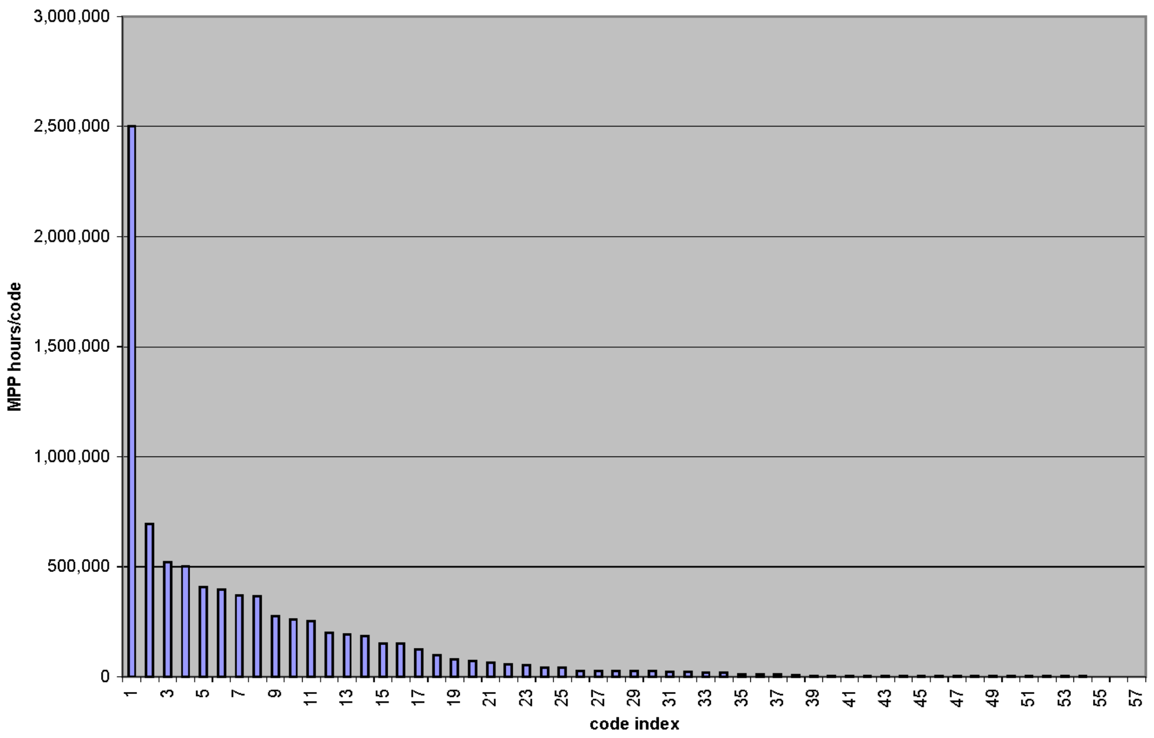

Fig.1, the MPP hours used on different codes in the chemical science category. This is a plot of the MPP column versus the index column in Table.1

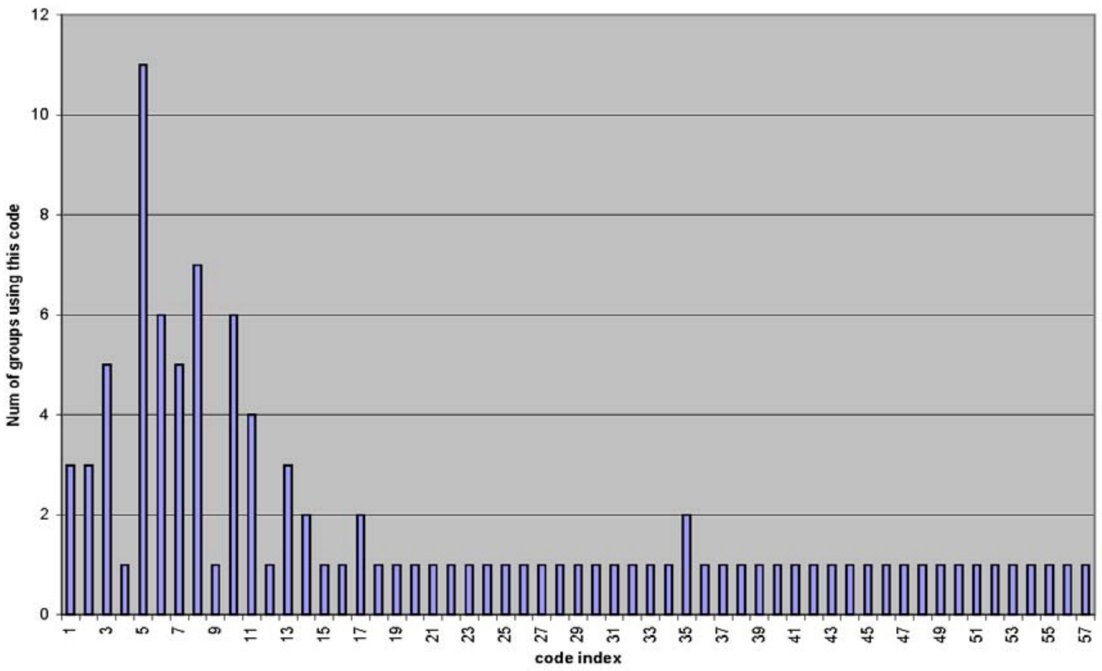

Fig.2, the number of groups using a particular code. This is a plot of column N-group versus the index column in Table.I.

In the following, we present the direct user survey results, conducted via email. We have survey 4 heavily used codes: Molpro, Gaussian, Zori, and DL_POLY. They represent traditional quantum chemistry calculations (Molpro and Gaussian), quantum Monte Carlo calculations (Zori) and classical molecular dynamics calculations (DL_POLY). We did not survey S3D, because S3D is from an INCITE project (which might change from year to year), and it calculates combustion which does not represent typical chemical science work load. Our choice of code is not only based on the MPP hours, but also based on the number of groups using it. We didn't survey all the planewave based DFT codes because they are traditionally covered by the material science category. We have asked six questions, from what are the science they did, to what are the executable code names. After this survey, we have also checked some of the NERSC machine blog file records, which have information for each executable run during a particular period of time. This information includes: the executable name, the number of processors used, and the total execution time. We find that our direct user survey results roughly agree with the machine blog record. From the survey results, we have the following summary: for the DFT/HF/MP2/CCSD codes: Molpro and Gaussian, the parallelization is not very good. Most people only use them on one node or even one processor. They run for 12-24 hours. For the quantum mechanical code Zori, because it is embarrassingly parallel, it has used from 64 processors to 2000 processors, depending on the availability of the processors. It can take 48 hours. For the classical molecular dynamics code: DL_POLY, it can run efficiently on 1024 processors for million atom systems. But in practice, it runs using 8 to 64 nodes for 500 hours (MD simulation). From this, we see that there is a demand for capacity computation and long time simulation in chemical science category. Even the seemly large parallel calculations (the Zori quantum Monte Carlo calculation) are actually capacity calculations because there is no communication between different processors (only the resulting energies from different processors are statistically averaged at the end of the simulation), although there could be a load balance issue when some of the walkers in MC algorithm die and new walkers are generated.

The survey questions and answers are listed below:

(1) What is the physical problem you are solving ? (e.g., molecular dynamics, atomic structure relaxation, reaction path, binding energy calculation, electronic structure, many body effects, DFT, MP3, CCSD,HF ...)

    Answers:

    (a) Molpro user1 (Bastiaan J. Braams): small system (mostly 7 atoms) electronic structure calculations, try to map out the potential functions. Need thousands of calculations for different molecular configurations.
    (b) Molpro user2 (Cheuk-Yiu Ng): We do the electronic structure calculations, mainly, using the DFT, MP2, CCSD(T) methods.
    (c) Molpro user3 (Andy Simmonett): CCSD(T) for anharmonic vibration.

(d) Gaussian user1 (Karma Sawyer): Structure relaxation, frequency and anharmonicity. Transition metals.

(e) Gaussian user2 (Ivan Mikhaylov): Transition dipole moments, geometry optimization, potential surface.

(f) Gaussian user3 (YiYang Sun): Binding energy (DFT, HF, MP2, CCSD).

(g) Zori user1 (Brian Austin): Molecular electronic structure using quantum Monte Carlo.

(h) Zori user2 (Victor Batista): Quantum molecular dynamics, atomic structure calculations.

(i) DL_POLY user1 (Patrick Redmill): Molecular dynamics, the free energy of solvantion for nanoparticle.

(j) DL_POLY user2 (Perla Balbuena): MD simulation to study the dynamics properties, segregation on bimetallic nanoparticles, phonon spectra, solvent effects.

(2) Which NERSC machine are you spending most time running the job? Do you plan to use the new Franklin machine?

Answers:

(a) Molpro user1 (Bastiaan J. Braams): exclusively on Jacquard (for cheap charging rate and turn around time).

(b) Molpro user2 (Cheuk-Yiu Ng): Jacquard and Bassi. Do plan to use Franklin.

(c) Molpro user3 (Andy Simmonett): The version on Jacquard has a bug. So use it on bassi. Run on a single node (slower on more nodes!).

(d) Gaussian user1 (Karma Sawyer): Jacquard. No plan for Franklin (don't know that machine).

(e) Gaussian user2 (Ivan Mikhaylov): Jacquard. No plan for Franklin (maybe).

(f) Gaussian user3 (YiYang Sun): Bassi. Yes, plan to use Franklin.

(g) Zori user1 (Brian Austin): Currently mostly on Seaborg. Yes, definitely plan to use Franklin.

(h) Zori user2 (Victor Batista): Using Jacquard, do plan to use Franklin.

(i) DL_POLY user1 (Patrick Redmill): Seaborg. Don't know about Franklin.

(j) DL_POLY user2 (Perla Balbuena): Seaborg.

(3) Most importantly, what are the typical number of processors and typical number of hours you are running the job?

Answers:

(a) Molpro user1 (Bastian J. Braams): 128 CPUs for 12 hours (on low queue). But they are actually serial jobs each using one node. Just many different jobs running in parallel.
(b) Molpro user2 (Cheuk-Yiu Ng): A few nodes on Jacquard for 48 hours, a single node on Bassi for 36 hours.
(c) Molpro user3 (Andy Simmonett): Single node (8 processors) for 8 hours.
(d) Gaussian user1 (Karma Sawyer): One processor for 5 to 100 hours.
(e) Gaussian user2 (Ivan Mikhaylov): 2 processors, 20 hours.
(f) Gaussian user3 (YiYang Sun): 8 processors (one node), 12-36 hours.
(g) Zori user1 (Brian Austin): 256 to 2025 processors, for the maximum allowed time. It is embarrassingly parallel.
(h) Zori user2 (Victor Batista): 48 processors, for 24-48 hours.
(i) DL_POLY user1 (Patrick Redmill): 8 processors (1 node), 500 hours per job.
(j) DL_POLY user2 (Perla Balbuena): 16-64 processors, 8 hours.

(4) Do you know what is the most time consuming part inside the code ? (e.g., what algorithm, what library routine, I/O, matrix construction (four body integral?), matrix diagonalization, FFT....).

Answers:

(a) Molpro user1 (Bastian J. Braams): not so clear. On the part of RCCSD(T), perhaps on the matrix integral, or solving the coupled cluster equation, or on some AMD linear algebra lib. The calculation can also be disk intensive.
(b) Molpro user2 (Cheuk-Yiu Ng): Heavily on Blas library.
(c) Molpro user3 (Andy Simmonett): The O(N^7) step in the T correction, perhaps using Blas (GotoBLas routines are faster than ATLAS or Intel's MKL library).
(d) Gaussian user1 (Karma Sawyer): Do not know.
(e) Gaussian user2 (Ivan Mikhaylov): 11002.exe. It solves the coupled perturbation HF, second order perturbation, etc.
(f) Gaussian user3 (YiYang Sun): Four body integral, and the CCSD equations (parallelization on this part is poor).
(g) Zori user1 (Brian Austin): Basis function evaluation, and matrix multiplication in the Slater determinate. Also the error evaluation takes time (repeated thousands of times).
(h) Zori user2 (Victor Batista): NAG library. Matrix multiplications, evaluation of determinants, computation of numerical integrals.
(i) DL_POLY user1 (Patrick Redmill): Reciprocal space Ewald summation.

(j) DL_POLY user2 (Perla Balbuena): Communication cost, the replica data arrangement needs a lot of communications.

(5) Given the increasing parallelization of the machine, do you plan to change the code? Change the algorithm? Do you think there could be limitation for your parallelization?

Answers:

(a) Molpro user1 (Bastian J. Braams): Parallel Molpro doesn't work well. So currently running the serial version. Not plan to use parallel version.
(b) Molpro user2 (Cheuk-Yiu Ng): The parallelization on Molpro is limited by the speed of interconnect.
(c) Molpro user3 (Andy Simmonett): Don't have the source code, not plan to improve the code. Parallelization is limited by interconnect.
(d) Gaussian user1 (Karma Sawyer): No plan (it is a commercial code).
(e) Gaussian user2 (Ivan Mikhaylov): No plan.
(f) Gaussian user3 (YiYang Sun): No plan.
(g) Zori user1 (Brian Austin): No limit for parallelization, because it is embarrassingly parallel, for many statistical runs. Only need to collect the data from different runs.
(h) Zori user2 (Victor Batista): No limit for parallelization, embarrassingly parallel.
(i) DL_POLY user1 (Patrick Redmill): No plan to change the code. No plan to use large number of processors anyway.
(j) DL_POLY user2 (Perla Balbuena): No plan to change the code. The code is already well parallelized.

(6) What is the name of your executable (so we can use automatic ways to get more information)?

Answers:

(a) Molpro user1 (Bastian J. Braams): mpiexec -n 128 $HOME/bin/bjb-mp-scan and /usr/common/usg/molpro/2002.6/bin/molpro
(b) Molpro user2 (Cheuk-Yiu Ng): Molpro
(c) Molpro user3 (Andy Simmonett): molprop
(d) Gaussian user1 (Karma Sawyer): Gaussian (?)
(e) Gaussian user2 (Ivan Mikhaylov): /usr/common/usg/g03/c2/g03/l1002.exe
(f) Gaussian user3 (YiYang Sun): g03
(g) Zori user1 (Brian Austin): Zori
(h) Zori user2 (Victor Batista): prog

(i) DL_POLY user1 (Patrick Redmill): DLPOLY.X

(j) DL_POLY user2 (Perla Balbuena): DLPOLY.X